

Home Exam – Software Engineering – D7032E - Reexam

Teacher: Josef Hallberg, josef.hallberg@itu.se, A3305

Instructions

- The home reexam is an individual examination.
- **The home reexam has no deadline.** Take as much time as you need.
 - The teacher makes no promises of quick assessment of your reexam.
- **Book at time for an oral exam-session** with Josef Hallberg when you are done.
 - Availability of oral exam-sessions depends on the teacher's schedule and workload.
- The submission should consist of a compressed file containing a pdf file with answers to the written questions and any diagrams/pictures you wish to include, and your code. Place all files in a folder named as your username before compressing it. You may find it difficult to attach compressed files directly in e-mails, in which case you can email a link to a file-sharing site where your compressed file is stored.
- Use subject **"D7032E: Home Exam"** so your hand-in won't get lost (I will send a reply by email within a few days if I've received it).
- It should contain **original** work. You are not allowed to cheat, in any interpretation of the word. You are allowed to discuss questions with classmates, but you must provide your own answers. Additionally, you are allowed to use AI tools and use the material produced by these tools, however, **what you hand in should represent your knowledge and understanding of the subject.** That means, if AI tools are generating answers or code for you, it is your responsibility to study the material provided by an AI and **make sure you understand the generated material, understand design decisions that were made by the AI model, and are able to justify and motivate design decisions.**
- Your external references for your work should be referenced in the hand in text. All external references should be complete and included in a separate section at the end of the hand in.
- The language can be Swedish or English.
- The examiner reserves the right to refuse to grade a hand-in that does not use a correct/readable language. Remember to spellcheck your document before you submit it.
- Write in running text (i.e. not just bullets) – but be short, concrete and to the point!
- Use a 12 point text size in a readable font.
- It's fine to draw pictures by hand and scanning them, or take a photo of a drawing and include the picture; however make sure that the quality is good enough for the picture to be clear.
- Judgment will be based on the following questions:
 - Is the answer complete? (Does it actually answer the question?)
 - Is the answer technically correct? (Is the answer feasible?)
 - Are selections and decisions motivated? (Is the answer based on facts?)
 - Are references correctly included where needed and correctly? (Not just loose facts?)
- **To have part B assessed and graded you need to motivate your design in a one-to-one oral exam session. The purpose of the oral exam session is to assess whether the material you have handed in represent your understanding of the subject. This is to make sure you do not hand in AI generated material, or material produced by other people, that you do not yourself understand. The examiner reserves the right refuse assessing part B in cases where the material you have handed in does not represent your understanding of the subject.**

Total points: 25	
Grade	Required points
5	22p
4	18p
3	13p
U (Fail)	0-12p

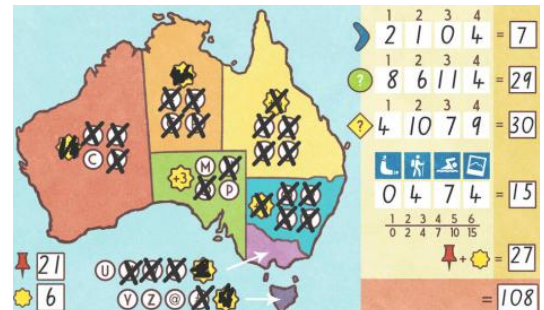
Good luck! /Josef

Scenario: Boomerang Australia

Boomerang Australia is a simple "draft & write" game for 2-4 players. Visit the most beautiful places of Australia. While touring, you will meet local animals and enjoy tourist activities, such as hiking, swimming or simply taking beautiful pictures! But don't forget that this trip is not only for pleasure — you all compete to be the best visitor and discover the most of Australia! Complete original rules are available as separate PDFs in Canvas.

- Short overview of Boomerang Australia / Europe / USA: https://www.youtube.com/watch?v=i_tY80QAE1Y
- Longer playthrough of Boomerang Australia: <https://www.youtube.com/watch?v=m7Tu8CFUkJ0>

AussieDev is a newbie developer who decided to turn Boomerang Australia into an online computer game and has future plans of expanding the game to support optional rules and the Boomerang Europe and Boomerang USA editions as well. Unfortunately, AussieDev isn't very good at software design and there are several bugs. Your role is to help AussieDev improve upon the design by following the remaining instructions in this test. You should also structure the design in such a way that new editions and optional rules can be added in the future (but you don't need to implement these changes, just make sure the design support future extensions and modifications according to best practices).



3. Shuffle the 28 cards.
4. Deal 7 cards to each player.
5. Each player selects a Throw card (this is not revealed to other players until the round ends)
 - a. The goal when selecting the Throw card is to secure a good Catch. This is done by calculating the difference between the number on the Catch card (the last card you receive) and the Throw card. The difference equals the number of points received (the absolute value of the difference).
6. Each player pass the remaining cards in their hand to the next player in the sequence (the last player passes cards to the first player)
7. Each player selects one card from their new hand and shows it to the other players.
8. Continue step 6 and 7 until only one card remain on hand. Show all cards selected in these steps to the other players.
9. The final card is passed to the previous player in the sequence (the first player passes the card to the last player). The card is added to that player's shown cards and is known as the Catch card.
10. Scoring:
 - a. *Throw and catch score*: Compare the number on your Throw (first) card and the Catch (final) card and calculate the difference (the absolute value): this is your Throw and catch score.
 - b. *Tourist sites score*: Each card also has a letter (A-Ö) corresponding to a site in Australia. The sites are divided into Australia's seven regions: Western Australia (A-D), Northern Territory (E-H), Queensland (I-L), South Australia (M-P), New South Wales (Q-T), Victoria (U-X), and Tasmania (Y-Z, *, -). Score one point for each site visited (the letters on the played cards this round) and note down which sites have been visited as they do not score points again in coming rounds, but they count towards completing a region (see below).
 - i. If you are the first player to complete a region you gain 3 bonus points
 - ii. If more than one player completes a region at the same time they all gain the 3 bonus points.
 - c. *Collections score*: Many cards also have a collection attribute. Each item has a value associated with it: Leaves are 1, Wildflowers are 2, Shells are 3, and Souvenirs are 5. Add up all the values of each item you collected on your cards this round.
 - i. If the total for your collections is 1-7 then you double that number as your score for that round.
 - ii. If your value is over 7, you only score your collection number as points.
 - d. *Animals score*: For each pair of matching animals on the cards you drafted this round, you score points marked on that animal (Kangaroos are 3, Emus are 4, Wombats are 5, Koalas are 7, and Platypuses are 9). Total what you score for all the matching animal pairs you collected this round.
 - e. *Activities score*: Scoring activities is optional. For any single activity you would like to score each round, count how many of that activity you have on your seven cards and score the corresponding points:

# Activity cards	1	2	3	4	5	6
Score	0	2	4	7	10	15

 - i. You may only score one Activity per round.
 - ii. You may only score each Activity once per game (collection of rounds).
11. The next round – at the end of each round, after everyone has finished scoring and the score is shown to the player, start a new round from step 2.
12. Game end – After scoring the fourth round, the game ends. Players add up all their scores in each category, including the region bonuses. The highest score wins.
 - a. In the case of a tie, the tied player who scored the most *Throw & Catch* points wins.

Future modifications: (you do not need to implement these but create your design for modifiability and extensibility)

13. To add more variation to the card drafting in a three or four player game, alternate passing cards to previous and then to the next player each round.
14. Support the game modes in Boomerang Europe and Boomerang USA (see separate PDF files with rules) in addition to Boomerang Australia. This involves new cards with new tourist sites and regions, new symbols (in addition to Collection, Animal, and Activity), and new scoring mechanisms.

Quality attributes: Design with priority on *Modifiability*, *Extensibility* (requirement 13 and 14), and *Testability*.

Questions

(page 4/5)

Part A**1. Unit testing**

(2p, max 1 page)

Which requirement(s) (rules and requirements 1 – 12 on previous page) is/are currently not being fulfilled by the code (refer to the requirement number in your answer)? For each of the requirements that are not fulfilled answer:

- If it is possible to test the requirement using JUnit without modifying the existing code, write what the JUnit assert (or appropriate code for testing it) for it would be.
- If it is not possible to test the requirement using JUnit without modifying the existing code, motivate why it is not.

2. Software Architecture design and refactoring

(9p, max 2 pages excluding diagrams)

- a. Identify shortcomings in the original design. Use SOLID principles and Booch's metrics in your reasoning (1p)

Consider the requirements (rules and requirements 1 – 12 and the quality attributes with requirements 13 – 14 in mind on the previous page) and the existing implementation. Update / redesign the software architecture design for the application. The documentation should be sufficient for a developer to understand how to develop the code, know where functionalities are to be located, understand interfaces, understand relations between classes and modules, and understand communication flow. Use good software architecture design and coding best practices according to SOLID principles and Booch's metrics.

- b. Reason about design choices in your **new** (refactored) design using SOLID and Booch (3p)
- c. Reason about how quality attributes have been satisfied in the **new** design. What design choices did you make specifically to meet these quality attribute requirements. Divide your reasoning into separate sections per quality attribute (helps me assess your exam) (3p)
- d. Motivate design patterns or choice not to use in your **new** design. If any, what purpose do they serve and how do they improve the design (2p)

Note: The oral exam-session will cover primarily your software architecture design. For this session you will be expected to answer questions about your design and motivate design choices. You are allowed to refer to diagrams and code provided in your hand-in. It may therefore be more important to produce supporting diagrams than lengthy texts, since motivations for design choices can be given verbally. You should be able to answer questions about your design and about design choices without having to look at the written text (but looking at diagrams is recommended).

Part B

(page 5/5)

3. Quality Attributes, Design Patterns, Documentation, Re-engineering, Testing

(14p)

Refactor the code so that it matches the design in question2 (you may want to iterate question 2 once you have completed your refactoring to make sure the design documentation is up to date). The refactored code should adhere to the requirement (rules and requirements 1 – 12 on previous page). Things that are likely to change in the future, divided into quality attributes, are:

- Extensibility: Additional game-modes, such as those described in the “Future modifications to the game” may be introduced in the future.
 - Modifiability: The way network functionality is currently handled may be changed in the future. Network features in the future may be designed to make the server-client solution more flexible and robust, as well as easier to understand and work with. In addition, the potential game modes (requirement 14) adds new types card types and scoring mechanisms, which will change how some operations are handled. Furthermore, an optional rule for alternating to which player the hand is passed may be introduced.
 - Testability: In the future when changes are made to both implementation, game rules, and game modes of the game, it is important to have established a test suite and perhaps even coding guidelines to make sure that future changes can be properly tested. You only need to make tests for requirements 1-12.
- a. To what degree can the game be extended (extensibility) in the future (requirement 6)? (2p)
 - b. To what degree can it be modified (modifiability) in the future (requirement 6)? (3p)
 - c. To what degree is the code designed for testability? (1p)
 - d. To what degree is the code unit-tested (requirement 1-5) (2p)
 - e. To what degree does the code follow best practices (structure, standards, naming, etc.) (1p)
 - f. To what degree does the game correctly implement the functionalities of the game (2p)
 - g. To what degree are errors/exceptions handled and reported appropriately? (1p)
 - h. Is the code appropriately documented? (1p)
 - i. Is the code true to the design in question 2 (1p)

Please help AussieDev by re-engineering the code and create better code, which is easier to understand. There is no documentation other than the comments made inside the code and the requirements specified in this Home Exam on page 2 and 3. The code and official game rules for the core game as well as additional game modes are available in Canvas

The source code is provided in one file, BoomerangAustralia.java which contains the server, the client, and the code for one player, as well as all the game-states and game logic. (For server - run with: `java BoomerangAustralia [#Players] [#Bots]` (example: `java BoomerangAustralia 2 0`) for client – run with: `java BoomerangAustralia [IP-address of server]` (example: `java BoomerangAustralia 127.0.0.1`)). The server must be started before any of the clients are started. The server waits for the online clients to connect before starting the game.

In the re-engineering of the code the server does not need to host a player and does not need to launch functionality for this. It is ok to distribute such functionalities to other classes or even to the online Clients. The essential part is that the general functionality remains the same.

Add unit-tests, which verifies that the game runs correctly (it should result in a pass or fail output), where appropriate. It is enough to create unit-tests for requirements (rules and requirements 1 – 12 and any of the additional future modifications that are implemented from requirements 13 – 14 on page 3). The syntax for running the unit-test should also be clearly documented. Note that the implementation of the unit-tests should not interfere with the rest of the design.

If you are unfamiliar with Java you may re-engineer the code in another structured programming language. However, instructions need to be provided on how to compile and run the code on either a Windows or MacOS machine (including where to find the appropriate compiler if not provided by the OS by default).

It is not essential that the visual output when you run the program looks exactly the same. It is therefore ok to change how things are being printed etc.